Lab 5_1 보고서

20210479 이주현

① 개요

Lab 5_1은 산술 논리 장치(Arithmetic Logic Unit, ALU)의 구조를 이해하고 이를 구현해보는 것을 목표로 한다.

② 이론적 배경

(1) Arithmetic logic unit (ALU)

ALU란 산술 장치와 논리 장치로 구성되어 여러 가지 연산을 수행하는 장치이다. 산술 장치에서는 사칙 연산을 비롯한 연산들을 수행하고, 논리 장치에서는 bitwise 논리 연산을 수행한다.

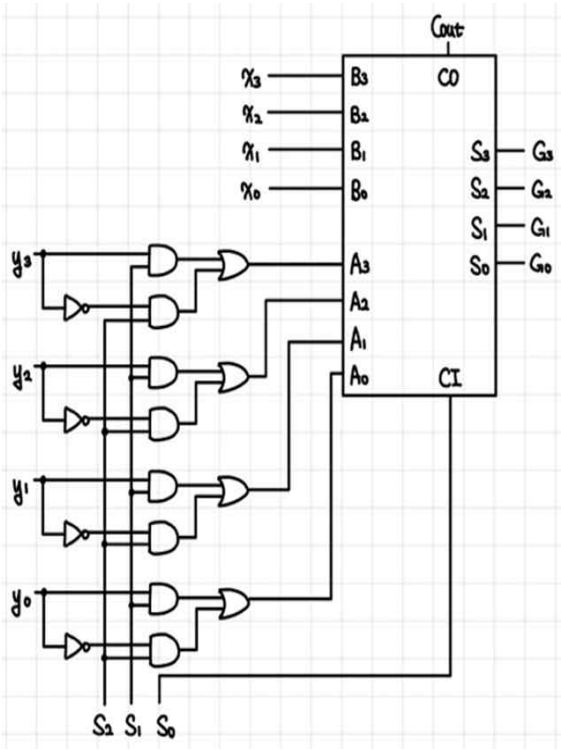
③ 실험 준비

ALU 중 산술 장치를 단순화하여 회로로 나타내면 아래와 같다.

S_2	S_1	y_i	A_i
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	1

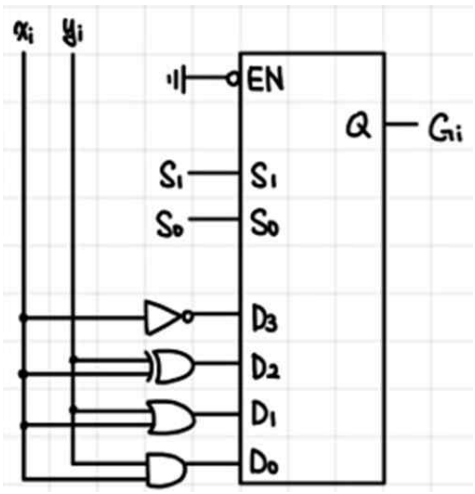
S_2 \ S_1y_i	00	01	11	10
0	0	0	1	0
1	1	0	1	1

$A_i = S_2y_i' + S_1y_i$

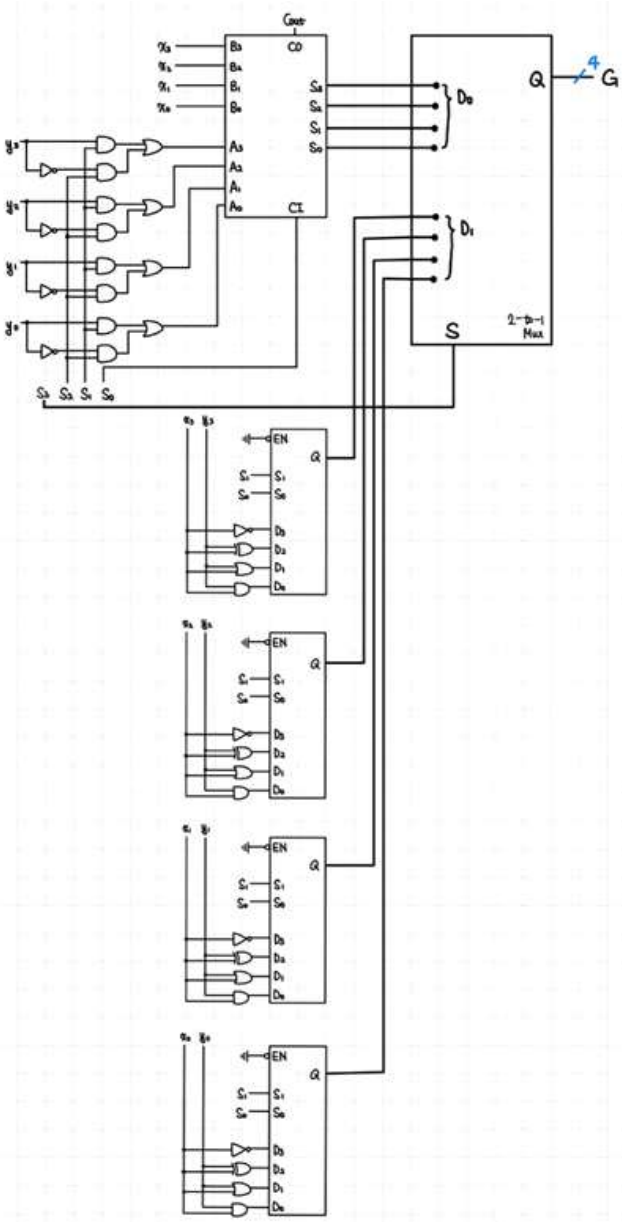


ALU 중 논리 장치를 단순화하여 회로로 나타내면 아래와 같다.

S_1	S_0	Output:
0	0	$G_i = x_iy_i$
0	1	$G_i = x_i + y_i$
1	0	$G_i = x_i \oplus y_i$
1	1	$G_i = x_i'$

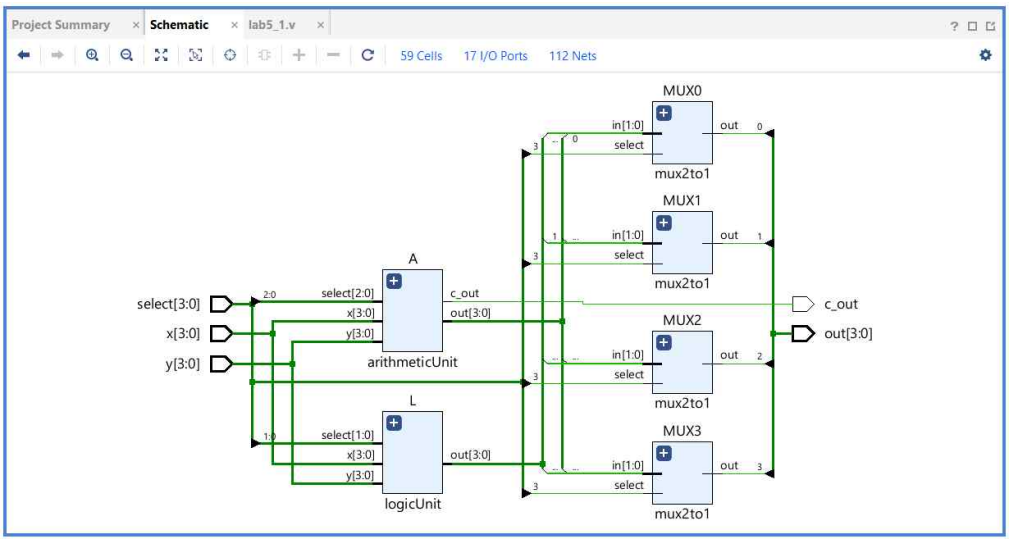


이 산술장치와 논리장치를 2-to-1 MUX로 묶어 ALU의 회로도를 구현하면 아래와 같아진다.

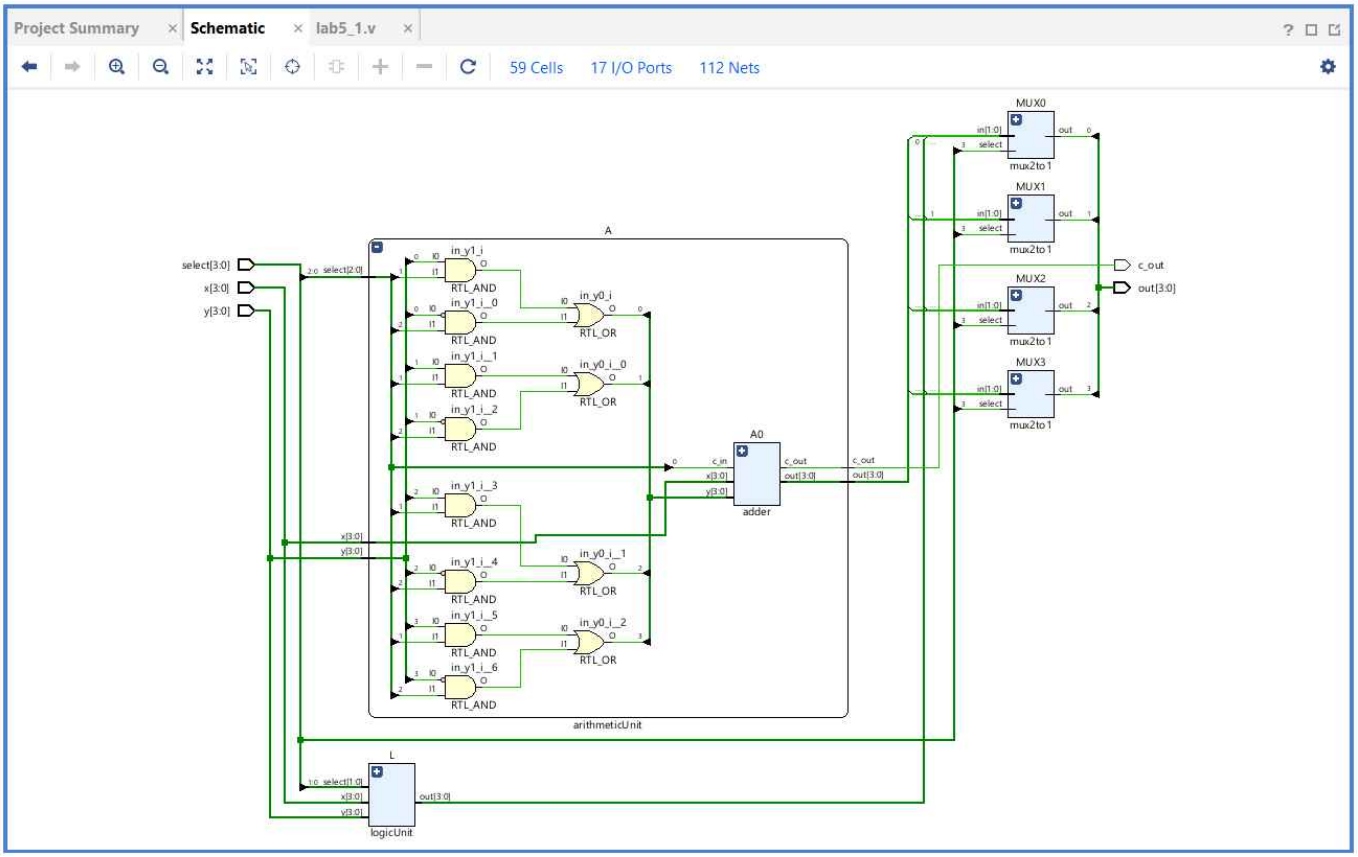


④ 결과

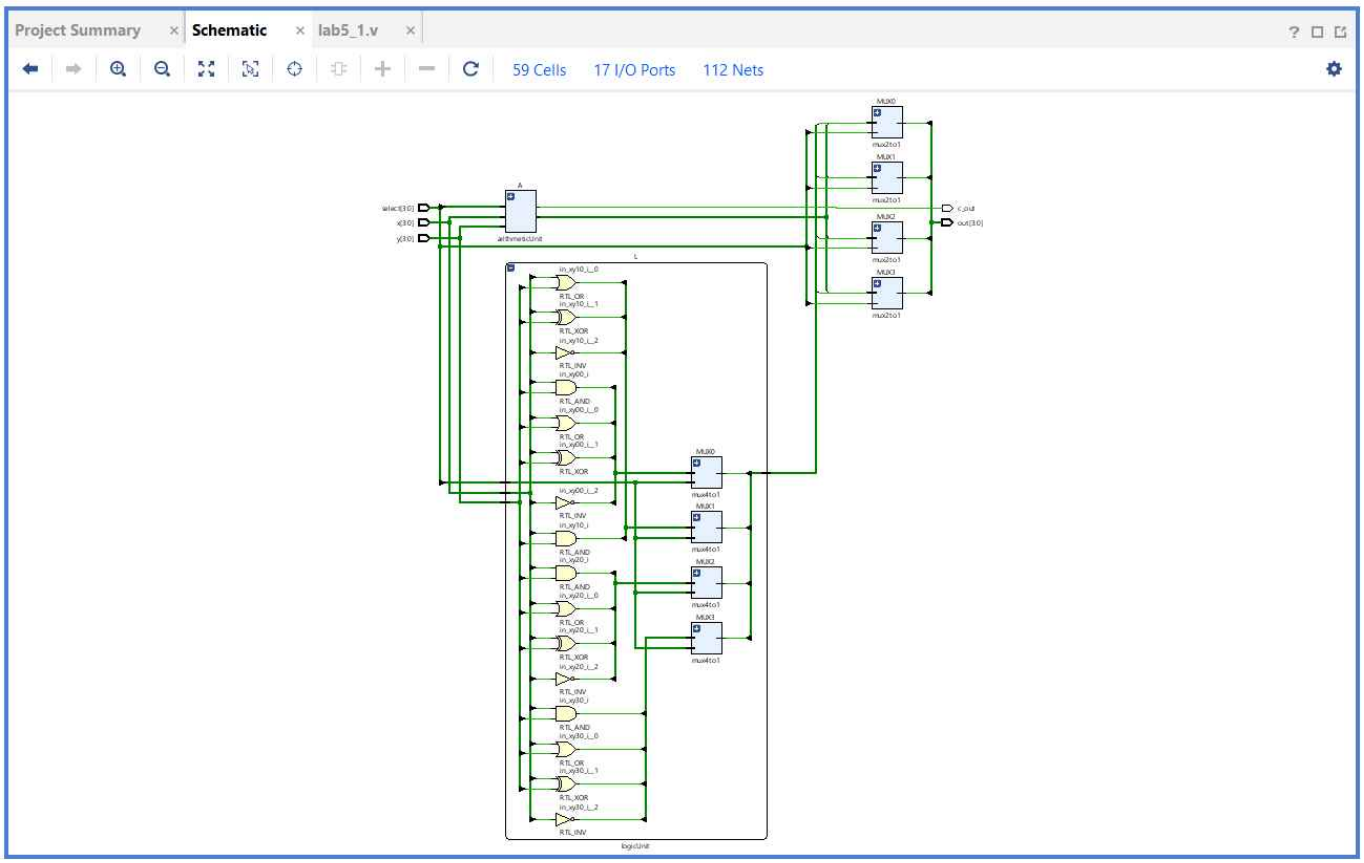
작성한 회로도(RTL analyzer schematic)는 아래와 같았다.



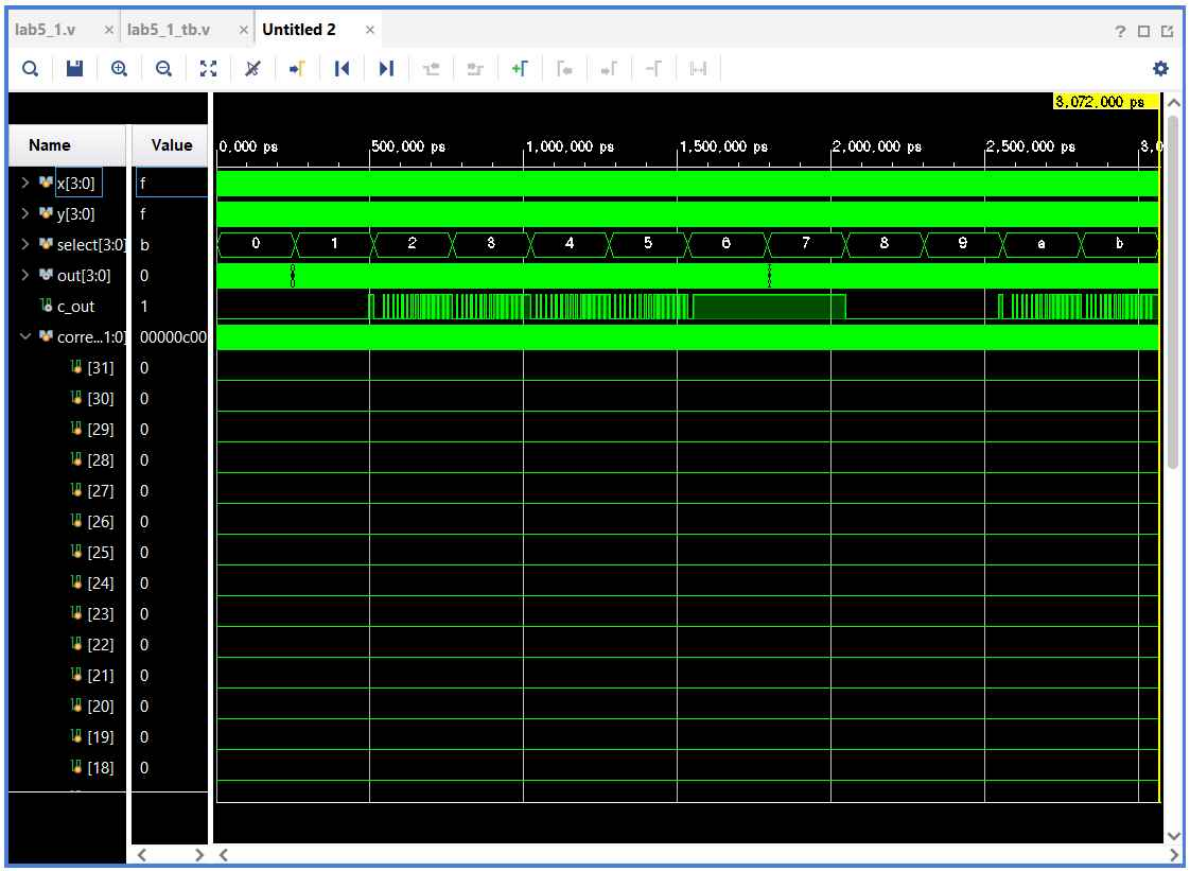
산술 장치 내부 회로도는 아래와 같다.

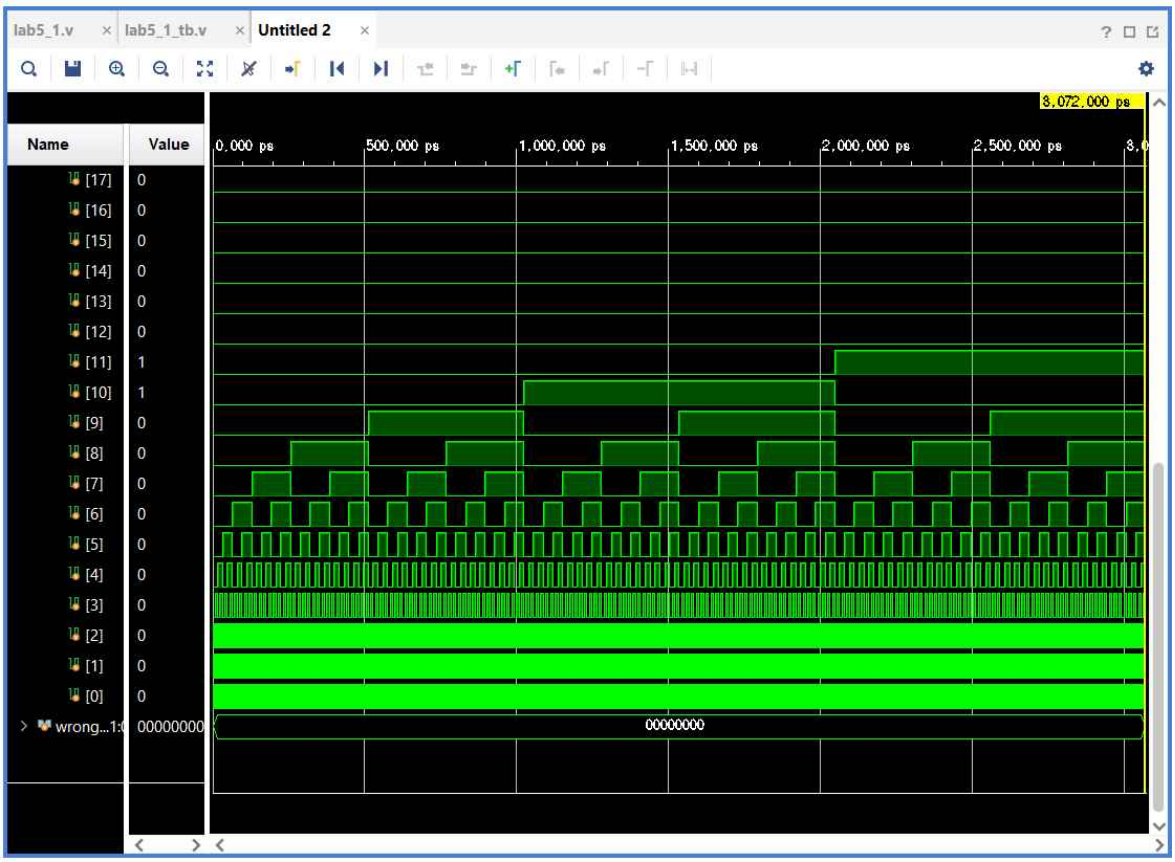


논리장치 내부 회로도는 아래와 같다.



Test bench를 작성하여 시뮬레이션을 돌린 결과는 아래와 같았다. 이를 통해 ALU의 구현이 올바르게 이루어졌음을 알 수 있다.





⑤ 논의

테스트벤치를 작성하는 데에 많은 어려움을 겪었으나, xvlog라는 txt 파일에 에러가 발생한 곳과 그 이유가 적혀 있어서 그것을 보고 참고하여 테스트벤치 작성을 완료할 수 있었다. 또한 실험 준비에서 그려놓은 회로도와 RTL 회로를 비교하며 구현이 올바르게 되었음을 알 수 있었고, log에 뜨는 error case 또한 테스트벤치 작성에 있어서 많은 도움이 되었다.

Lab 5_2 보고서

20210479 이주현

1 개요

Lab5_2는 정보 저장에 이용되는 JK flip flop의 개념과 회로의 구성에 대해 학습한 뒤, 이를 직접 구현해 보는 것을 목표로 한다.

2 이론적 배경

(1) JK latch & JK flip flop

JK latch는 SR latch에 추가 회로를 더해 SR latch의 한계를 개선한 장치이다. 그러나 J와 K가 동시에 1일 경우 race condition이 일어난다는 새로운 문제를 갖는다. 이는 latch가 입력값의 변화가 출력값의 변화로 직결되는 비동기 회로이기 때문이다. 그러나 이러한 toggle 현상은 위험한 경우가 많다.

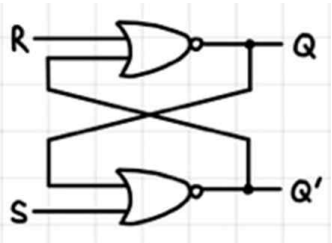
이를 해결하기 위해 flip flop이후 FF로 통칭)을 이용하는데, FF란 특정 신호에 따라 다른 회로와 어떤 특정한 순간에 맞추어 값을 변화시키는 동기 회로이다. JK FF는 clock 신호를 받아 이에 맞추어 작동한다는 특징을 갖는다.

(2) Master-slave JK FF

Master-slave JK FF는 SR latch 두 개를 연결하여 만든 FF이다. 주기적으로 값이 바뀌는 clock을 추가로 입력받아 clock이 1인 동안 master latch가 활성화되며, 0인 동안 slave latch가 활성화되는 동기 회로이다.

3 실험 준비

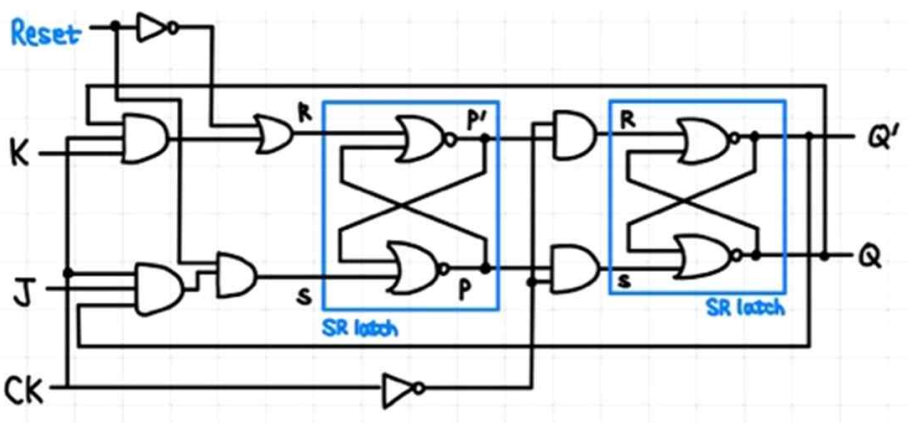
우선 SR latch의 회로를 그려보면 아래와 같다.



이를 바탕으로 negative reset master-slave JK FF의 회로를 구성해 보았다. 초기 입력으로 받는 J, K의 값을 J, K, 실제 master latch에 들어갈 J, K값을 J', K'라 하여 truth table을 그려보았다. 이를 바탕으로 reset pin과 gate를 추가하여 아래와 같은 negative reset master-slave JK FF를 구성하였다.

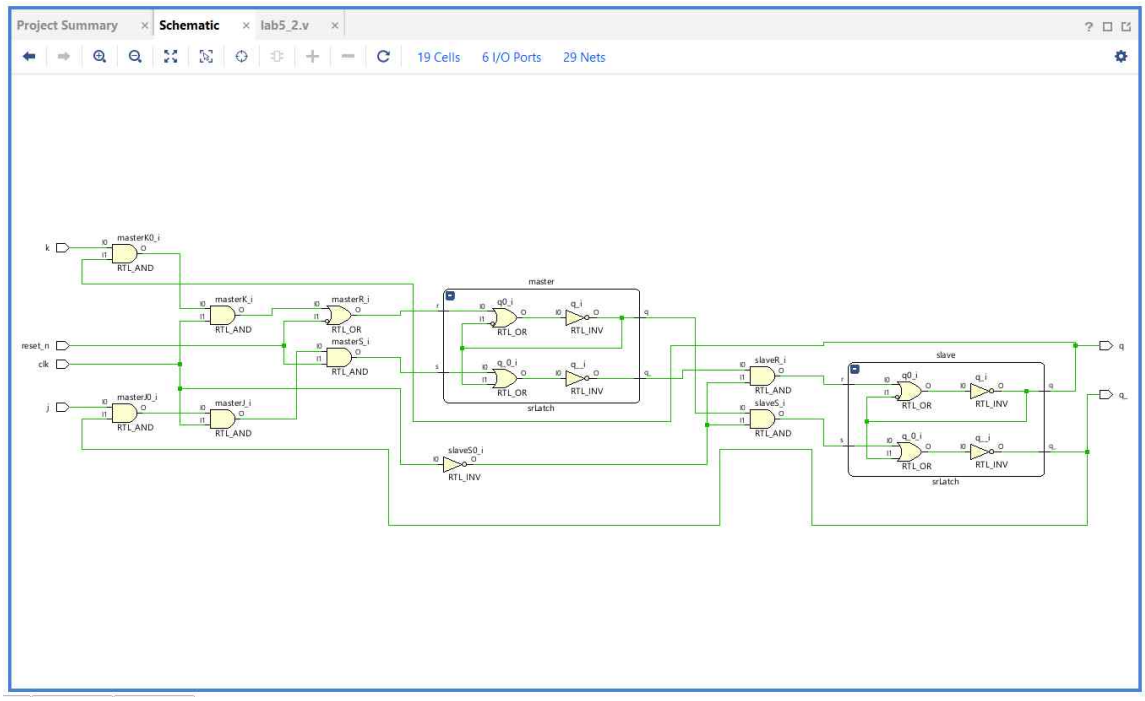
Reset	J	K	J'	K'
0	0	0	0	1
0	0	1	0	1
0	1	0	0	1
0	1	1	0	1
1	0	0	0	0
1	0	1	0	1
1	1	0	1	0
1	1	1	1	1

$J' = Reset \cdot J,\ K' = Reset' + K$



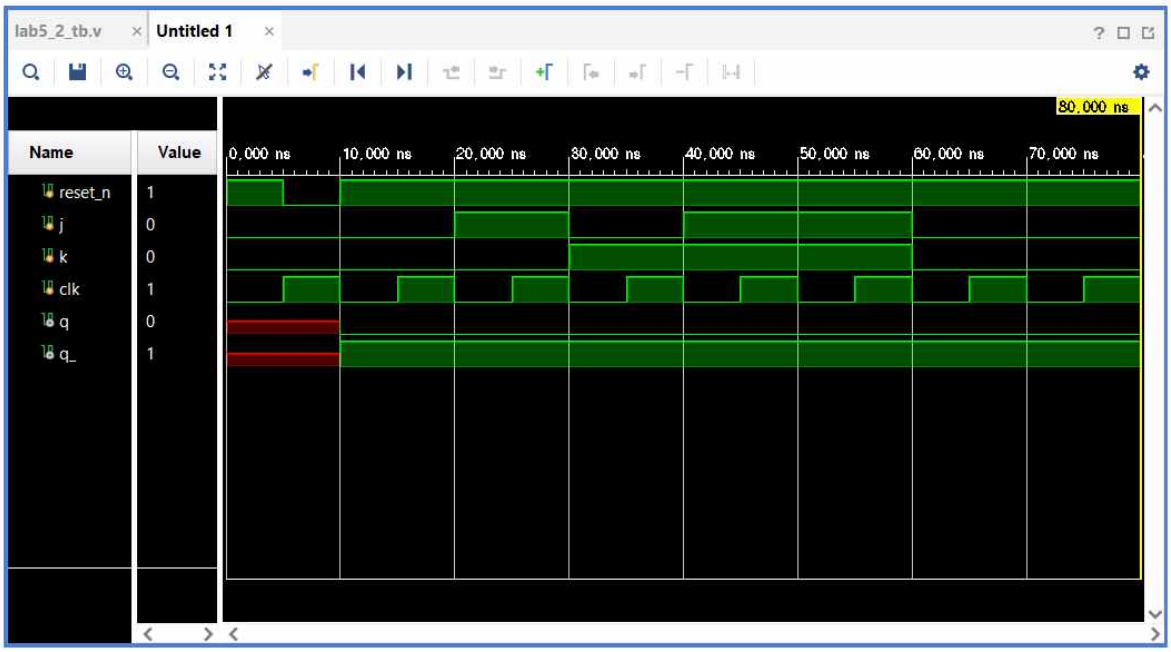
이론적 배경을 학습하며 보았듯 master-slave FF는 동기 회로로, 어떤 latch가 활성화되는지 여부는 clock의 값에 달려 있다. 따라서 clock의 값에 따라 활성화되지 않은 상태의 latch에 glitch가 발생하는 경우 그것에 영향을 받지 않을 수 있다. 그러나 활성화 상태의 latch에 glitch가 발생하는 경우에는 취약하다는 단점이 있다.

④ 결과



Negative reset master-slave JK FF를 구현한 회로도(RTL analyzer schematic)는 아래와 같다. 사각형으로 표시된 부분이 SR latch이다. Master-slave JK FF이 SR latch를 이용하여 잘 구현되었음을 확인할 수 있다.

테스트벤치를 작성하여 시뮬레이션을 실행한 결과는 아래와 같다.



⑤ 논의

Master-slave JK FF의 개념과 구현 방법에 대해서 알 수 있었다. 다만 테스트벤치 작성 미숙으로 인해 master-slave JK FF의 정상 작동 여부를 판단하기는 어려웠다. 실험 준비에서 예측한 해결 가능한 glitch를 구분해 내는 것 또한 테스트벤치 작성 미숙으로 인해 정확히 확인하지 못하였다.